SaaS-yrityksen tuotehallinta

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

HAMK Visamäki, 25.3.2010

Tommi Partanen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Työn nimi	SaaS-yrityksen tuotehallinta
Tekijä	Tommi Partanen
Ohjaava opettaja	Lasse Seppänen
Hyväksytty	_____._____.20_____
Hyväksyjä	

HAMK Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Tommi Partanen **Vuosi** 2010

Työn nimi SaaS-yrityksen tuotehallinta

TIIVISTELMÄ

Työn toimeksiantaja Movenium Oy on työhön liittyvän tiedon keräämiseen erikoistunut yritys, joka tarjoaa pääohjelmistotuotteitaan palveluina SaaS (Software as a Service) – mallin mukaisesti. Yrityksen tuotteiden ominaisuuksista saattavilla olevat tiedot olivat liian hajallaan ja epämääräisiä, jotta niitä olisi voitu hyödyntää tehokkaasti. Työn tarkoituksena oli selvittää, minkälainen tietokantaratkaisu tarvitaan asiakaskohtaisten tuotetietojen keräämiseen ja hallintaan toimeksiantajan sovellusalustassa.

Työssä sovellettiin teorioita, jotka liittyvät Saas-malliin sekä relaatiotietokantojen suunnitteluun. Tutkimusaineistoa kerättiin tutkimalla toimeksiantajan olemassa olevaa järjestelmää sekä keräämällä lisätietoja kyselyillä sekä haastatteluilla. Opinnäytetyössä käytettiin tutkimus- ja tiedonhankinta menetelmänä laadullista ja määrällistä tutkimusmenetelmää.

Työn päätuloksena toimeksiantaja sai tietokantaratkaisun, joka palvelee yrityksen eri osastoja sovellusalustaan liittyvien tietojen ylläpidossa. Ratkaisun pohjalta on mahdollista tehdä käyttöliittymäsovelluksia. Lisäksi kerätyn tutkimusaineiston avulla toimeksiantaja sai selville, mitä tietoja yritys tarvitsee tuotteiden ominaisuuksista toiminnan tehostamiseksi.

Avainsanat Relaatiotietokannat, Software as a Service, sovellusvuokraus, tuotehallinta.

Sivut 31 s. + liitteet 22 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Unit
Name of degree programme
Option

Author Tommi Partanen **Year** 2010

Subject of Bachelor's thesis Product management of SaaS oriented company

ABSTRACT

Client of this of this thesis is Movenium Oy. The company specializes in collecting work related data. Movenium offers its main software products as services according to the Software as a Service business model. Information about the features of the company's products was too too indefinite in order to use this information efficently. The purpose of this thesis was to find out what kind of database solution would be needed for collecting and managing customer-related product features on Movenium's platform.

Theories related to SaaS-model and relational database design were applied in this thesis. Research material was collected by Movenium's current system as well as by inquiries and interviews. Both qualitative and quantitative research methods were used.

As the main conclusion, client will be provided with a database solution which serves the different branches of the company in managing product features. In addition, the database solution enables to develop various user interfaces. With the help of the collected research material the client is also able to determine which product feature data is needed to improve the efficiency of the operations.

Keywords Relational databases, Software as a Service, software leasing, product management.

Pages 31 p. + appendices 22 p.

SISÄLLYS

Käsitteet

Movenium Platform

Movenium Platformilla tarkoitetaan toimeksiantajan kehittämää PHP- ja MySQL-pohjaista sovellusalustaa. Sen sisällä toimii Movenium Työajanseuranta, joka tunnettiin aiemmin tuotenimellä Leimaus. Tämä nimi esiintyy edelleen esimerkiksi tietokannan nimeämiskäytännöissä.

Lisäksi sovellusalustaan kuuluvat tuotteet Movenium Ajopäiväkirja ja Movenium Kalustonhallinta. Movenium Platformiin viitataan työssä sanalla *sovellusalusta*. Yrityksen sisällä siihen viitataan usein lyhenteellä MPF.

Ominaisuus

Sovellusalustaan liittyviin asetuksiin ja ominaisuuksiin viitataan työssä sanalla *ominaisuus.* Mainitut ominaisuudet ovat Movenium Platformiin ohjelmoituja asetuksia, joita yrityksen henkilökunnan ja jossain määrin myös asiakkaiden on mahdollista konfiguroida. Sovellusalustan ominaisuudet mahdollistavat siihen kuuluvien palveluiden räätälöinnin asiakaskohtaisesti. Toimeksiantajan sisäisessä kommunikoinnissa ominaisuuteen saatetaan viitata myös sen englanninkielisellä vastineella *feature*.

1 JOHDANTO

Termi SaaS eli *Software As a Service* on vakiintunut muutaman viime vuoden aikana suomalaiseen informaatioalan sanastoon. Tähän termiin sekä siihen liittyviin toimintamalleihin viitataan alan piirissä ahkerasti, mutta todellinen tieto tästä ohjelmistoliiketoiminnan mallista on hajallaan ja osittain epäselvää jopa alan yrityksissä työskenteleville. Näin on myös opinnäytetyön tekijän kohdalla.

Aihe tukee opinnäytetyön tekijän omia työtehtäviä toimeksiantajayrityksen asiakaspalvelussa ja ohjelmistokehityksessä. Aihe tarjoaa mahdollisuuden tutustua SaaS-malliin, sen tuottamiin haasteisiin ja näiden haasteiden ratkaisumahdollisuuksiin.

Toimeksiantaja Movenium Oy on todennut, että yrityksen SaaS-mallin mukaisesti tarjottujen ohjelmistotuotteiden ja niiden ominaisuuksien hallinta on liian raskasta. Tämä vaikuttaa koko yrityksen toimintaan ja voi hidastaa strategisesti tärkeiden tavoitteiden toteutumista. Tuotetietoja olisi tarkennettava ja niiden hallintaa parannettava, jotta yritys voi tehostaa toimintaansa.

Työ pyrkii vastaamaan kolmeen tutkimusongelmaan. Ensinnäkin toimeksiantaja haluaa tietää, minkälaisia ominaisuuksia ohjelmistotuotteisiin liittyy. Toiseksi on selvitettävä, mitä tietoja yrityksen eri osastot tarvitsevat tuotteista. Kolmanneksi toimeksiantaja tarvitsee tietokantaratkaisun, jonka avulla tuotetietojen keräys ja hallinta on mahdollista asiakaskohtaisesti toimeksiantajan sovelluksissa.

Työn toimeksiantaja, Movenium Oy, on liikkuvan työn mobiiliratkaisuihin erikoistunut yritys. Yrityksen perustajat aloittivat toiminnan tarjoten erilaisia internetpalveluja yksityishenkilöille ja yrityksille. Ensimmäinen palvelu luotiin vuonna 2002. Osakeyhtiömuotoiseksi yritys muuttui vuonna 2007 sähköisen työajanseurannan kasvatettua liiketoimintaa.

Tällä hetkellä yrityksessä on 15 työntekijää, joista osa tekee työtään etätyönä. Movenium on erikoistunut yrityksille suunnattuihin ohjelmistoihin ja ohjelmistopalveluihin. Näihin kuuluvat konsultointi, koulutuspalvelut, yrityksille palveluina tarjottavat ohjelmistot sekä erilaiset projektitoteutukset kuten järjestelmäintegraatiot.

2 TUTKIMUKSEN KOHDE

2.1 Toimeksiantajan sovellusalusta ja tietokanta

Sovellustalusta on rakennettu PHP-ohjelmointikielellä. Lyhenne tulee sanoista *PHP: Hypertext Preprocessor* (http://php.net, 2009) . PHP on avoimeen lähdekoodiin perustuva luokkakirjaston sisältävä ohjelmointikieli, jota käytetään erityisesti web-palvelinympäristöissä. PHP on komentosarjakieli, jossa ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa. Ohjelmointikieltä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä.

Alustan tietokanta on toteutettu MySQL-järjestelmällä. MySQL on erityisesti internetpalveluissa suosittu SQL-relaatiotietokannan hallintajärjestelmä. Sitä kehittää ruotsalainen yritys MySQL AB, jonka omistaa Sun Microsystems. MySQL perustuu avoimeen lähdekoodiin, mutta siitä on saatavana myös kaupallinen versio (MySQL AB, 2009).

Toimeksiantaja tarjoaa internetin kautta käytettäviä palveluita, joissa ohjelmistotuote ja siihen tallennetut tiedot voivat sijaita hajautetusti eri palvelimilla. Tällaisesta arkkitehtuurista käytetään nimitystä pilviarkkitehtuuri, joka on yksinkertaisimmillaan vain toinen ilmaisu internetille (Donaldson 2009, Techwatch Syyskuu 2009). Pilviarkkitehtuurissa käyttäjä ei välttämättä tiedä, missä hänen käyttämänsä data fyysisesti sijaitsee.

Sovellusalusta toimii Linux-käyttöjärjelmässä ylläpidetyllä Apache-palvelimella. Apache HTTP Server on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma, joka on Apache Software Foundation säätiön avoimen lähdekoodin tuote (Apache Software Foundation, 2009).

Ohjelmistotuotanto on jaettu kahdelle palvelimelle: kehityspalvelimelle ja tuotantopalvelimelle. Ohjelmistosuunnittelijat tekevät tuotekehitystä kehityspalvelimella, jossa sijaitsee versionhallintaohjelmisto. Sovellusalustaa käyttävien asiakkaiden palvelut sijaitsevat tuotantopalvelimella.

Näiden kahden palvelimen välillä on joitakin eroavaisuuksia esimerkiksi PHP:n asetuksissa, mutta nämä erot eivät vaikuta tietokannan sisältöön, eivätkä siten opinnäytetyöhön. Molemmilla palvelimilla on oma tietokantansa, jota sovellusalusta käyttää. Tuotantopalvelimen tietokanta kopioidaan joka yö kehityspalvelimelle, jotta kehityspalvelimen käyttämät tiedot ovat ajan tasalla testausta varten. Kaikki työssä käytetyt kaaviot ja kuvat toimeksiantajan nykyisestä tietokantajärjestelmästä on luotu kehityspalvelimen tietokannasta ellei toisin ole mainittu.

Sovellusalustaan liittyvät tiedostot voidaan jakaa karkeasti kolmeen luokkaan Model-View-Controller –arkkitehtuurin mukaisesti. Malli (Model) huolehtii järjestelmän tiedon tallentamisesta, ylläpidosta ja käsittelystä.

Näkymä (View) määrittää käyttöliittymän ulkoasun ja mallin tietojen esitystavan käyttöliittymässä. Ohjain tai kontrolleri (Controller) vastaanottaa käyttäjältä tulevat käskyt sekä muuttaa mallia ja näkymää vastauksena niihin.

Sovellusalusta ei kuitenkaan historiansa ja PHP:n luonteen vuoksi noudata MVC-arkkitehtuuria täysin, vaikka jotkin sen osat näin toimivatkin. Järjestelmässä yksittäinen tiedosto voi hoitaa jopa kaikkia arkkitehtuurin osia sisältäen käyttöliittymän osia, tiedon käsittelyn tai jopa luokan sekä tarvittavat funktiot käyttäjän vastaanottaman informaation käsittelyyn.

Suurin osa sovellusalustasta koostuu edelleen yksittäisistä PHP-tiedostoista ilman luokkapohjaista rakennetta. Tämä on mahdollistanut nopean tuotekehityksen ja palveluiden asiakaskohtaisen räätälöinnin hyvinkin yksityiskohtaisella tasolla, mutta samaan aikaan järjestelmän kasvu on jättänyt puutteita sovellusalustan ominaisuuksista tarjolla oleviin tietoihin.

Puutteisiin voidaan löytää useita syitä. Toimeksiantajalta puuttuu kattavat sisäiset käytännöt ja määrittelyt sille, mitä tietoja ohjelmistosuunnittelijoiden olisi annettava sovellusalustaan kehitetyistä uusista ominaisuuksista. Tuotekehitystä on tehty nopeasti asiakkaiden toiveiden mukaisesti, jolloin ominaisuuksien suunnittelulle, hallinnalle ja dokumentoinnille ei ole jäänyt tarpeeksi aikaa. Tämä on toki myös taloudellinen kysymys resurssien muodossa. Sovellusalusta ja tietokanta mahdollistavat yhdessä tyhjien tietokenttien tallentamisen. Ilman luokkarakennetta toimivat yksittäiset tiedostot ovat mahdollistaneet sen, että sovelluksien toimintaa on voitu tarkastella ja muuttaa helposti kooditasolla ilman kattavaa dokumentointiakin. Asiakaskohtainen tieto ominaisuuksista on ollut taloudellisesti tärkeämpää kuin yrityksen sisäinen tieto ominaisuuksista. Tietokantaan ei ole ollut mahdollista tallentaa kaikkia tarvittavia tietoja, koska tarvittava rakenne on puuttunut.

2.2 Työn rajaukset

Tuotehallinta oli rajattu relaatiotietokannan rakenteen ja sisällön suunnitteluun taulurakenteen tasolla. Tarvittaessa voitiin antaa ohjelmointiesimerkkejä järjestelmän hyödyistä, mutta tarkoituksena ei ollut esittää valmiita ohjelmointi- tai käyttöliittymäratkaisuja.

Työssä ei puututtu toimeksiantajan sovellusten versionhallintaan eli siihen, kuinka sovellusten kehitystyötä ja päivittämistä hoidetaan järjestelmässä. Vaikka versionhallinta liittyikin osittain joihinkin työn osa-alueisiin ja versionhallinta saattaa tulevaisuudessa vaikuttaa myös työn lopputuloksena esitettyyn tietokantaratkaisuun, ei aihetta ollut tarpeellista käsitellä tässä yhteydessä laajemmin. Versionhallinnan käytäntöihin olisi saattanut tulla muutoksia toimeksiantajan taholta opinnäytetyön valmistumisen aikana.

Työssä suunnitellun tietokantaratkaisun mahdollistamiin ohjelmistoihin tai niiden käyttöliittymiin otettiin kantaa vain yleisellä tasolla ilman konkreet-

tisia esimerkkejä. Työn ulkopuolelle jäivät kaikki järjestelmän palvelinlaitteistoihin liittyvät asiat mukaan lukien niiden ylläpito. Työn teknisenä pohjana käytettiin toimeksiantajan PHP- ja MySQL-ohjelmistoihin perustuvaa palvelinta, mutta mainittuja tekniikoita käsiteltiin ohjelmistosuunnittelun eikä ylläpidon näkökulmasta.

3 TUTKIMUSMENETELMÄT

Valituilla tutkimusmenetelmillä suoritettiin laadullista eli kvalitatiivista tutkimusta ja määrällistä eli kvantitatiivista tutkimusta. Kerätty laadullinen aineisto sisältää tekstimuotoista tietoa. Määrällinen aineisto sisältää numeromuodossa esitettyä tietoa. Määrällinen aineisto voidaan esittää tilastollisilla menetelmillä.

Jaottelu laadullisiin ja määrällisiin aineistoihin on perusteltua siksi, että laadullisen aineiston keräämiseen ja analysointiin käytettävät menetelmät poikkeavat määrällisen aineiston keräämiseen ja analysointiin käytettävistä menetelmistä. Laadullista aineistoa saatiin tutkimuksessa haastattelemalla ja tutkimalla olemassa olevia dokumentteja sekä järjestelmiä. Määrällisiä aineistoja saatiin kyselylomakkeella ja olemassa olevista tietokannoista löytyviä tietoja tutkimalla.

Tutkimuksien pääpaino oli laadullisen aineiston keräämisessä, sillä yrityksen pienen henkilöstömäärän takia tarpeeksi luotettavan määrällisen aineiston hankkiminen esimerkiksi kyselyjä käyttäen oli vaikeaa. Laadullisella tutkimuksella selvitettiin olemassa olevan järjestelmän rakenne, yrityksen ja sen henkilöstön tarpeet sekä henkilöstön mielipiteitä sovellusalustan ominaisuuksiin liittyen.

3.1 Nykyisen tietokannan kartoitus

Tietokannan jatkokehittämistä varten on ymmärrettävä sen toiminta tällä hetkellä. Nykyisen järjestelmän kartoitus oli välttämätöntä tämän ymmärryksen saamiseksi. Tietokannan tutkinta toimi laadullisen ja määrällisen aineiston hankintamenetelmänä. Tutkinta voidaan käsittää olemassa olevien dokumenttien tutkinnan korvaavana tutkimuksena, koska lähtötilanteessa dokumentaatiota tietokannasta ei ollut olemassa.

3.2 Haastattelut

Haastattelu on tiedonhankinnan perusmuoto, jota käytetään kerättäessä tietoa ihmisten mielipiteistä, ajatuksista ja uskomuksista. Tällöin kerätään laadullista aineistoa. Haastattelua voidaan käyttää tiedon hankkimiseen silloin kun tietoa ei ole dokumentoitu, vaan se on esimerkiksi siirtynyt suullisesti sukupolvelta toiselle. Haastattelussa ollaan suorassa kielellisessä vuorovaikutuksessa tutkittavan kanssa. Haastattelu koetaan useimmiten ihmisläheiseksi ja metodeiltaan miellyttäväksi tiedonhankintamenetelmäksi sekä tutkijan että tutkittavan kannalta. Menetelmänä haastattelu on joustava ja sopii siksi monenlaisiin tutkimuskohteisiin. (Hirsjärvi & Hurme, 2009.)

Haastattelulla on tutkimusmenetelmänä sekä etuja että haittoja. Etuna on se, että ihminen voi olla tutkimuksen aktiivinen osapuoli ja toimia vuorovaikutuksessa haastattelijan kanssa. Vähän kartoitettua aluetta voidaan

tutkia helpommin, koska tällaisissa tapauksissa tutkijan on vaikea tietää oikeita kysymyksiä tai vastausten suuntia ennakkoon. Haastatteluilla voidaan tarkentaa olemassa olevaa tietoa ja selventää tai tarkentaa aiemmin saatuja vastauksia. (Hirsjärvi & Hurme, 2009.)

Haittapuolena on, että haastattelu ja sen valmistelut voivat viedä runsaasti aikaa. Haastattelua ei aina voida pitää täysin luotettavana tietolähteenä, koska sen osapuolet vaikuttavat tuloksiin. Tämä voi tapahtua myös tahattomasti. Vapaamuotoisten tulosten tulkitseminen ja purkaminen on usein ongelmallista ja aineistoa saattaa kertyä suuria määriä. (Hirsjärvi & Hurme, 2009.)

3.3 Kyselyt

Kysely ja kyselylomakkeet ovat tutkimushaastattelun muoto, jolla pyritään saamaan tilastollisesti mitattavaa määrällistä tietoa. Kyselyillä voidaan halutessa kerätä myös tietyissä määrin laadullistakin aineistoa. Tyypillisesti kyselyitä käytetään esimerkiksi mielipidekyselyissä tai tilanteissa, jolloin halutaan kerätä dokumentoimatonta tietoa tietyissä viitekehyksissä. Kyselyillä voidaan siis rajata tarkkaan rajat joiden sisällä kyselyyn osallistuvat antavat tietoa. Kyselyt sopivat hyvin myös ei-tieteelliseen tiedon keruuseen. (Hirsjärvi & Hurme, 2009.)

Toisin kuin haastattelussa, kyselytutkimuksessa ei välttämättä vaadita haastattelijan tai tutkijan läsnäoloa tutkimusta tehtäessä vaan tutkittava voi antaa vastauksensa ennalta laadittujen ohjeiden mukaan. Tämä vapauttaa tutkijan itse tiedonhankkimisprosessista, mutta tuo mukanaan sen ongelman, ettei tutkija voi tarkentaa vastauksia haastateltavalta. Kyselytutkimus vaatiikin tutkijalta ymmärrystä siihen, mitä kannattaa kysyä.

Lomakemuotoisten kyselyiden avulla voidaan kerätä tietoa huomattavasti suuremmalta määrältä tutkimuskohteita kuin pelkillä haastatteluilla. Tosin on huomioitava, että liian pieni otanta tutkittavasta ryhmästä saattaa vääristää saatuja tuloksia. Kyselylomakkeiden vastauksia on mahdollista prosessoida nopeasti ja rutiininomaisesti. (Hirsjärvi & Hurme, 2009.)

4 SOFTWARE AS A SERVICE -MALLI

Software as a Service (SaaS) –malli on ohjelmistoliiketoiminnan malli, jossa ohjelmistotuotteita tarjoava yritys antaa asiakkaalleen käyttöoikeuden verkkopalveluna tarjoamaansa ohjelmistoon käyttömaksua vastaan. SaaS-mallissa ohjelmisto tai palvelu otetaan käyttöön keskitetysti palvelutarjoajan palvelimella tai verkossa, johon asiakas voi ottaa yhteyden esimerkiksi internetin, sisäverkon tai VPN-yhteyden avulla. Asiakkaille annetaan käyttöoikeus ohjelmistoon, jota he eivät itse omista (SIIA 2001, 4-5).

Mallista käytetään joissain yhteyksissä myös nimitystä ohjelmistovuokraus. Palveluntarjoajaan sekä ohjelmistovuokrauksen liiketoimintamalliin saatetaan viitata termillä Application Service Provider (ASP). Software as a Service on kuitenkin yleistermi, joka pitää sisällään kaikki kuvatun ohjelmistovuokrausmallin mukaiset toimijat, toimintatavat ja liiketoimintamallit. Käytännöllä on pyritty selkiyttämään ohjelmistoalan terminologiaa. (SIIA 2001, 4-5.)

Tunnettuja esimerkkejä mallia käyttävistä yrityksistä ovat muun muassa myynnin ja liiketoiminnan verkkopalveluina tarjoava Salesforce.com, Google omilla verkon yli käytettävillä ohjelmistoillaan Google Apps sekä tiedostonjakopalvelut Box.net ja Dropbox.com. SaaS-mallia voidaan soveltaa kaupallisesti lähes mihin tahansa verkon kautta toimivaan sovellukseen.

4.1 Hinnoittelu

SaaS-mallin hinnoitteluperusteet voidaan jakaa karkeasti viiteen eri vaihtoehtoon, jotka on esitetty alla. Hinnoittelumalleja voidaan kuitenkin yhdistellä toisiinsa tai muokata tarpeen sekä tehdyn palvelusopimuksen mukaan.

1. *Tilauspohjainen veloitus:*
Asiakkaalta veloitettavat maksut lasketaan sen mukaan, mitä ohjelmistoja asiakas käyttää. Veloitus tehdään yleensä käyttäjämäärien mukaan. (SIIA 2001, 11.)

2. *Käyttömäärään perustuva malli:*
Veloitus tapahtuu ohjelmiston käytön perusteella ja liittyy tyypillisesti ohjelmiston ylläpidon vaatimien resurssien kuten palvelimien määrään. Veloitus voidaan sitoa esimerkiksi asiakkaan palvelun ylläpitämiseen käytettyjen palvelinprosessorien määrään ja käyttöasteeseen tai yhtäaikaisten käyttäjien määrään palvelussa tietyllä aikavälillä. (SIIA 2001, 11.)

3. *Transaktioihin perustuva malli:*
Asiakasta veloitetaan ohjelmiston transaktioiden eli tapahtumien määrän mukaan. Esimerkiksi palveluna tarjottavan laskutusohjelman käytöstä voi-

daan veloittaa tietty summa jokaista luotua laskua vastaan. Veloitus voi perustua myös tallennettujen tietojen määrään, käyttäjien tekemiin toimiin tai lähes mille tahansa lukumäärällisesti laskettavalle toiminnalle ohjelmiston sisällä. (SIIA 2001, 11.)

4. *Arvopohjainen tai jaetun riskin ja tuoton malli:*
Veloitus on suorassa suhteessa ohjelmiston avulla tavoiteltaviin liiketoiminnan tavoitteiden toteutumiseen. Esimerkiksi myynnin kontaktoinnin työkaluksi tarjotun palvelun veloitus voisi perustua asiakkaan myyntiosaston saavuttamien kontaktien määrään. Mitä enemmän palvelusta on hyötyä liiketoiminnalle, sitä suurempia palvelumaksut ovat. (SIIA 2001, 11.)

5. *Kiinteän veloituksen malli:*
Asiakkaalta perittävä veloitus palvelun käytöstä on kiinteä, tietyin aikavälein laskutettava maksu, joka voi perustua ohjelmiston tuettuun käyttäjämäärään, ohjelmiston ominaisuuksiin, tarjotun asiakaspalvelun ja tuen tasoon tai muihin palvelun hinnoitteluperusteisiin. (SIIA 2001, 11).

4.2 Mallin edut

Palvelun tarjoavalle yritykselle SaaS-malli antaa perinteistä ohjelmistoliiketoimintaa enemmän liikkumavaraa ja hallintaa asiakkaisiin nähden. Ohjelmisto voidaan päivittää ja ylläpitää keskitetysti. Vikatilanteiden selvittäminen on useimmissa tapauksissa helpompaa kuin esimerkiksi asiakkaan tiloihin päätteille asennettujen ohjelmistojen vikatilanteiden selvittäminen, koska päätelaitteiston vaikutus ohjelmiston toimintaan on pienempi.

Asiakkaille on mahdollista tarjota yksityiskohtaisemmin heidän tarpeisiinsa sopivia ratkaisuja ilman merkittäviä laiteinvestointeja. Usein SaaS-mallilla toimivat yritykset tarjoavat peruspalvelun, joka on tarvittaessa räätälöitävissä asiakkaan tarpeiden mukaan siihen on ostettavilla lisäosilla.

SaaS-mallilla toimiva yritys voi määritellä hyvinkin tarkkaan liiketoimintansa rajat. Hoidetaanko myynti itse vai jälleenmyyjien kautta? Ulkoistetaanko palvelinten ylläpito? Halutaanko myydä tiettyjä valmiita tuotteita vai jokaiselle asiakkaalle erikseen räätälöityjä palveluita? Äärimmillään vietynä SaaS-mallissa ohjelmistoyritys voi tuottaa ohjelmistoja, joiden tarjonnan, myynnin ja ylläpidon hoitavat muut tahot.

Asiakkaalle SaaS-mallin mukainen ohjelmistovuokraus on useimmiten kustannustehokas tapa käyttää omia tietotekniikkaresursseja. Palveluina tarjottavat ohjelmistot eivät normaalisti vaadi mittavia asennuksia, laiteinvestointeja tai ylläpitovastuuta asiakkaalta. Palveluiden käyttö voidaan järjestää jo olemassa olevan laitekannan avulla ja tietotekniikkaresurssit käyttää muihin tarkoituksiin. Lisäksi ohjelmiston päivittäminen on palveluntarjoajan vastuulla, eikä yleensä vaadi teknisiä toimenpiteitä asiakkaalta.

Monet SaaS-mallin mukaisesti tarjotut palvelut ovat alustariippumattomia, jolloin palvelun käyttö voidaan nähdä myös kilpailuttamista ja toiminnan

tehostamista helpottavana tekijänä. Asiakas ei ole sidottu tiettyyn laitteistoarkkitehtuuriin, verkko-operaattoriin tai henkilöstöön, jolloin asiakas voi halutessaan lopettaa palvelun käytön suhteellisen nopeasti verrattuna perinteiseen ohjelmistoliiketoimintaan, jossa olemassa olevat resurssit tai niiden puute tuo ylimääräisiä vaatimuksia ohjelmiston käyttämiselle yrityksessä. Ohjelmistovuokraus mahdollistaa ohjelmistotuotteiden käytön tarkemman seurannan, erityisesti jos tilatun palvelun laskutus on käyttömäärään tai transaktioihin perustuvan mallin mukainen.

4.3 Mallin haasteet

On itsestään selvää, että luotettavaa SaaS-mallilla tuotettua palvelua ei pystytä tarjoamaan, jos verkkoinfrastruktuuri ei ole sen edellyttämällä tasolla. Palvelua tuottavalle yritykselle korostuu yhteistyökumppaneiden merkitys. Jos ulkoistettu palvelinten ylläpito ei toimi toivotulla tavalla tai asiakkaan verkkoliikenteessä on häiriöitä, joutuu palvelun tarjoaja ensimmäisenä vastuuseen, vaikka vikatilanne ei siitä johtuisikaan. Tämä vaatii luottamusta ja sujuvaa kommunikaatiota yhteistyökumppaneiden kanssa.

Asiakkaiden erilaiset käyttötarpeet ja –tilanteet, näiden ymmärtäminen sekä tarpeiden tyydyttäminen ovat olennainen osa onnistunutta liiketoimintaa. Asiakaskohtaisesti räätälöitävien palveluiden kohdalla asiakkaiden tarpeet voivat teettää ylimääräistä työtä. Ymmärretäänkö asiakkaan tarpeet oikein ja pystytäänkö niihin vastaamaan olemassa olevilla palveluilla? Onko asiakas ymmärtänyt palvelun luonteen, rajoitukset ja mahdollisuudet oikein?

Verkon ja asiakkaan laitteistosta johtuvat virhetilanteet tuottavat huomattavan lisähaasteen palvelun tarjoajalle. Ohjelmistosta tai palvelusta itsestään johtuvat vikatilanteet ovat SaaS-mallissa usein suhteellisen helposti selvitettävissä, mutta asiakkaan laitteistosta johtuvat eivät välttämättä ole. Onko vika asiakkaan yhteydessä? Onko asiakkaan käyttämällä operaattorilla vikatilanne? Onko vika paikallinen?

Pilviarkkitehtuurilla toteutettujen palvelujen yhteydessä voidaan törmätä kansainväliseen lainsäädäntöön liittyviin epäselvyyksiin. Palvelussa käytetty data voi olla tallennettuna hajautetusti eri puolilla maailmaa sijaitseviin palvelinkeskuksiin. Minkä maan lainsäädäntöä sovelletaan, jos viranomaiset pyytävät luovuttamaan tietylle palvelimelle tallentuneita tietoja tai jos tietoja katoaa? Voiko valtio, jossa palvelin sijaitsee, evätä tietojen luovutuksen oman lainsäädäntönsä nojalla tai esimerkiksi valtion turvallisuuden perusteella?

Edellä mainitut kysymykset liittyvät omalta osaltaan myös asiakkaan kokemiin ongelmiin. Kuka näkee palveluna tarjottavaan ohjelmaan tallentamani tiedot? Miten ne varmennetaan? Tietoturva, sen hoitaminen ja toiminnan läpinäkyvyys voivat olla haasteita sekä asiakkaalle että palvelun tarjoavalle yritykselle.

Asiakkaalle palveluiden ja ohjelmistojen vertailu voi muodostua vaikeaksi niin ominaisuuksien kuin hinnoittelunkin osalta. SaaS-mallissa kustannusten vertaileminen voi olla erittäin vaikeaa jopa saman palvelun eri versioiden välillä. Asiakas saattaa kokea, että hänellä on liian monta vaihtoehtoa ja mahdollisuutta. Asiakkaan olisi myös sisäistettävä SaaS-malli niin, että palvelun toimintaperiaate verkon yli tarjottavana palveluna kaikkine mahdollisuuksineen ja rajoituksineen ei jää epäselväksi.

5 TUOTETIETOJEN KEHITTÄMINEN

Sovellusalustassa oli lähtötilanteessa noin 590 eri ominaisuutta, joiden määrä lisääntyi viikoittain. Toimeksiantaja oli todennut, että sovellusalustan ominaisuuksista saatavilla olevat tiedot olivat riittämättömät tai huonosti saatavilla. Tuotannon henkilöstölle tämä tuotti ylimääräistä työtä, koska ominaisuuksien toimintaa jouduttiin tarkastelemaan kooditasolla. Dokumentaation ja ohjeistuksen puuttuessa eri henkilöt saattoivat tulkita ominaisuuksien toimintaa eri tavoilla.

Tiedon puute vaikeutti myös kaupallisen puolen toimintaa. Henkilöstöllä ei ollut saatavilla tarpeeksi selkeää ja ajan tasalla olevaa tietoa sovellusalustan ominaisuuksista, jolloin markkinointi ja myynti eivät voineet käyttää kaikkea tuotteiden potentiaalia hyödyksi. Lisäksi tietokannasta puutui selkeä jako esimerkiksi eri tuotteita ja tuoteversioita koskeviin ominaisuuksiin.

Joissakin tapauksissa tarkat tiedot ominaisuuden toiminnasta olivat ainoastaan tekijällä itsellään. Tietokantaan ei tallennettu tietoa ominaisuuden tekijästä, jolloin tiedon hukkumisen riski kasvoi entisestään. Yrityksellä ei myöskään ollut sisäistä ohjeistusta siitä, mitä tietoja uusista sovellusalustaan lisättävistä ominaisuuksista tulisi tallentaa. Järjestelmä salli puutteellisten tietojen tallentamisen, jolloin ne jäivät helposti vajaiksi jo olemassa olevien tietokantakenttienkin osalta. Yrityksen henkilöstöllä oli pelko siitä, että sovellusalustan kehitys kaatuu sen omaan monimutkaisuuteensa, koska tiedon puutteen takia kaikkia ominaisuuksia ei osata tai pystytä ottamaan huomioon uusia kehitettäessä.

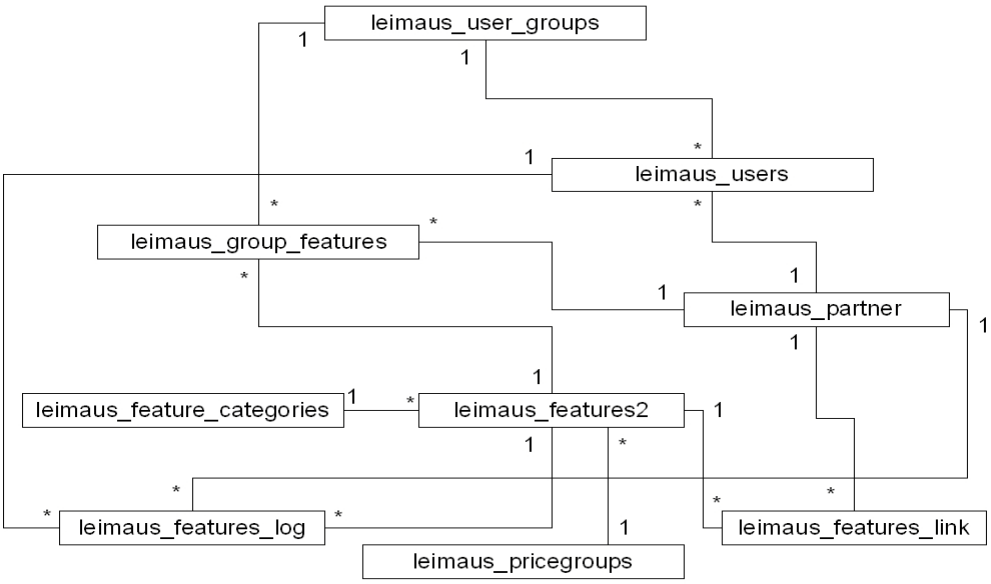
5.1.1 Nykyisen tietokannan kartoitus

Tietokantaa tutkittaessa ei voitu käyttää tietokantakaavioita tai kuvauksia, koska niitä ei ollut olemassa. Tietokannan rakennetta tutkittiin SQLyog nimisen asiakassovelluksen avulla. *Kuvassa 1* on esitetty kartoituksen perusteella tietokannasta löydetyt sovellusalustan ominaisuuksiin liittyvät kantataulut ja niiden suhteet toisiinsa. Taulujen kenttien kuvaukset ja viiteavaimet on esitetty *liitteessä 4*.

Taulu *leimaus_features2* sisältää varsinaiset sovellusalustan ominaisuudet, joille voidaan asettaa asiakaskohtaisia arvoja. Asiakkaiden tiedot löytyvät *leimaus_partner* taulusta ja asiakaskohtaiset ominaisuuksien arvot taulusta *leimaus_features_link*. Nämä kolme taulua muodostavat sovellusalustan ominaisuuksien ytimen.

Muut taulut tarjoavat lisätietoja ominaisuuksille. Lokitiedot muutoksista tallennetaan tauluun *leimaus_features_log*. Hinnoitteluluokat ominaisuuksille löytyvät *leimaus_pricegroups* taulusta. Ominaisuudet voidaan luokitella *leimaus_feature_categories* taulusta löytyvien kategorioiden mukaan. Ominaisuuksia voidaan tietyissä tapauksessa asettaa käyttäjäryhmäkohtai-

sesti. Nämä tallennetaan *leimaus_group_features* tauluun. Käyttäjien tiedot ovat taulussa *leimaus_users* ja käyttäjäryhmät taulussa *leimaus_user_groups*.



KUVA 1 *Sovellusalustan ominaisuuksiin liittyvät tietokantataulut ja niiden yhteydet.*

Kartoituksen yhteydessä selvitettiin myös tietokannan normalisoinnin tasoa. Normalisointimallin avulla relaatiotietokannan rakenne pyritään muotoilemaan niin, että se tukee tietojen ehjää tallennusta. Malli vähentää tiedon redundanssia eli saman tiedon tallentamista useaan kertaan ja parantaa tallennetun tiedon eheyttä.

Normaalimuotojen säännöt voidaan jakaa kolmeen eri vaiheeseen. Ensimmäisen normaalimuodon sääntö: ”Poista toistuvat ryhmät ja moniarvoiset sarakkeet.” Toisen normaalimuodon sääntö: ”Jos taulussa on moniosainen perusavain, niin kaikkien sarakkeiden tulee olla funktionaalisesti riippuvia koko perusavaimesta (eikä siis vain osa-avaimesta).” Kolmannen normaalimuodon sääntö:
”Jokaisen sarakkeen pitää olla funktionaalisesti riippuvainen vain perusavaimesta (eikä siis mistään ”tavallisesta” sarakkeesta)”. (Hovi, Huotari, Lahdenmäki, 2005.)

Sovellusalustan tietokannan tietorakenne ei täytä täysin minkään normaalimuodon sääntöjä. Tämä johtunee siitä, että tietojen käsittelyn kannalta sovellusalustassa esimerkiksi moniarvoiset sarakkeet ovat helpommin käsiteltävissä kuin erilliset tietokantataulut. Normalisointiin ei ilmeisesti ole ollut tarvetta puuttua, sillä sen puute ei ole toistaiseksi aiheuttanut ongelmia.

5.1.2 Tietokanta ja sovellusalusta

Tietokannan muutokset ja ominaisuuksien lisäys hoidetaan muokkaamalla sovellusalustan tiedostoa *db_updates.php*. Kaikki tietokantaan tehtävät rakennemuutokset sekä uusien ominaisuuksien lisäys hoidetaan tämän tiedoston kautta.

Tiedostossa voidaan käyttää kahta funktiota: *sql_lause()* ja *feature()*. Näillä funktioilla voidaan lisätä erilliselle ajolistalle tietokantaan ajettavia SQL-lauseita. Molemmissa funktioissa on käytössä samat parametrit: *funktion_nimi(Tietokannan juokseva versionumero, Mantis-tunniste, Kommentti, Ajettava SQL-lause)*. Funktiota *feature()* käytetään uusien ominaisuuksien lisäämiseen *leimaus_features2* tauluun. Funktiota *sql_lause()* käytetään muiden SQL-lauseiden ajamiseen kantaan.

Parametreissä esiintyvä *Mantis-tunniste* viittaa Mantis Bug Tracker ohjelmaan. Se on ilmainen avoimen lähdekoodin ohjelmistotuotannon työkalu, jolla voidaan hallita kehityspyyntöjä ja virheraportteja (Mantis Bug Tracker, 2009). Mantis Bug Tracker on käytössä toimeksiantajalla, joskin siitä ollaan pyrkimässä eroon korvaavien järjestelmien avulla.

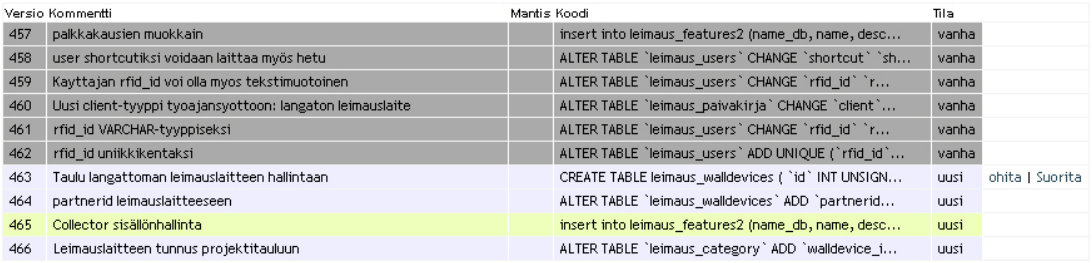
Seuraavassa esimerkissä on lisätty ominaisuus nimeltä *Selitteiden järjestely* käyttämällä funktiota *feature()*.

```
feature(527, 0000, "Selitteiden järjestely", "INSERT INTO `leimaus_features2` (`name_db`, `name`, `description`, `type`, `value_options`, `category`, `value_text`, `pricegroupid`, `timestamp`, `version`, `categoryid`, `show_level`) VALUES ('description_ordertool', 'Selitteiden järjestelytyökalu', 'Voidaan vaihtaa selitteiden järjestys drop-down-valikoissa', 'checkbox', '', 'general', '', '0', '2010-01-15 13:00:00', '3.1', '12', '0');
");
```

Seuraavaa esimerkissä on lisätty tauluun *leimaus_features2* uusi varchar-tyyppinen kenttä *link* käyttämällä funktiota *sql_lause()*.

```
sql_lause(530,0000,"Esimerkkimuutos","ALTER TABLE `leimaus_features2` ADD `link` varchar(256)");
```

Järjestelmän hallintaoikeudet omistava käyttäjä voi ajaa ajolistalta funktiokutsuja, jolloin halutut muutokset tehdään tietokantaan. Sovellusalustan käyttöliittymässä ajolista näkyy listana, joka on esitetty *kuvassa 2*. Kun listalta ajetaan muutos tietokantaan, päivittyy funktiokutsussa annettu tietokannan versionumero tauluun *leimaus_db*.



Versio	Kommentti	Mantis	Koodi	Tila	
457	palkkakausien muokkain		insert into leimaus_features2 (name_db, name, desc...	vanha	
458	user shortcutiksi voidaan laittaa myös hetu		ALTER TABLE `leimaus_users` CHANGE `shortcut` `sh...	vanha	
459	Kayttajan rfid_id voi olla myos tekstimuotoinen		ALTER TABLE `leimaus_users` CHANGE `rfid_id` `r...	vanha	
460	Uusi client-tyyppi tyoajansyottoon: langaton leimauslaite		ALTER TABLE `leimaus_paivakirja` CHANGE `client`...	vanha	
461	rfid_id VARCHAR-tyyppiseksi		ALTER TABLE `leimaus_users` CHANGE `rfid_id` `r...	vanha	
462	rfid_id uniikkikentaksi		ALTER TABLE `leimaus_users` ADD UNIQUE (`rfid_id`...	vanha	
463	Taulu langattoman leimauslaitteen hallintaan		CREATE TABLE leimaus_walldevices (`id` INT UNSIGN...	uusi	ohita | Suorita
464	partnerid leimauslaitteeseen		ALTER TABLE `leimaus_walldevices` ADD `partnerid`...	uusi	
465	Collector sisällönhallinta		insert into leimaus_features2 (name_db, name, desc...	uusi	
466	Leimauslaitteen tunnus projektitauluun		ALTER TABLE `leimaus_category` ADD `walldevice_i...	uusi	

KUVA 2 *Sovellusalustan SQL-lauseiden ajolista. Jo ajetut lauseet on merkitty harmaalla, uudet SQL-lauseet vaalean lilalla ja uudet ominaisuudet vaalean vihreällä värillä.*

5.1.3 Haastattelujen toteutus

Haastattelu soveltuu hyvin tuntemattoman tai vähän kartoitetun alueen tietojen keräämiseen (Hirsjärvi & Hurme, 2009). Haastattelu valittiin tutkimusmenetelmäksi, koska osa yrityksen sovellusalustaan liittyvistä tarpeista olivat selvittämättä tai niitä mietittiin yrityksen sisällä.

Työn lopputuloksen oli tarkoitus edesauttaa suoraan tai välillisesti sekä yrityksen teknistä että kaupallista kehittymistä, joten laadun varmistamiseksi oli saatava näkemyksiä molemmista näkökulmista. Haastattelut mahdollistavat tiedonkeruutilanteessa vapaan keskustelun sekä ryhmähaastattelut. Tällöin jo haastattelutilanteessa voidaan saada suoria vastauksia siihen, mitkä tiedot sovellusalustan ominaisuuksista ovat oleellisia yrityksen kannalta.

Ainoastaan yrityksen johtohenkilöillä oli tiedossaan tarkat tiedot yrityksen tulevaisuuden suunnitelmista. He tekevät päätökset työn tulosten käytännön sovelluksista. Haastatteluihin valittiin toimitusjohtaja Tuomo Pentikäinen, kehitysjohtaja Mikko Tuominen, innovaattori Sauli Karhu ja tuotekehitysjohtaja Veikko Salminen. Valituista henkilöistä Sauli Karhu oli henkilökohtaisista syistä estynyt osallistumaan haastatteluihin, joten tuotekehitykseen liittyviin kysymyksiin vastasi yksin Veikko Salminen.

Haastattelut tehtiin Skype-sovelluksen välityksellä. Skype on vertaisverkkoihin perustuva pikaviestintäohjelma, jonka avulla on mahdollista käydä ääni- ja videopuheluja, jakaa tiedostoja ja lähettää viestejä internetin välityksellä (Skype Limited, 2009). Ohjelma valittiin, koska tekstimuotoinen keskustelu on helppo dokumentoida ja ohjelma oli jo toimeksiantajan sisäisessä käytössä. Ääneen käydyn keskustelun ei katsottu tuovan lisäarvoa kerättävälle materiaalille. Tekstimuodossa käytävällä keskustelulla voitiin välttää Hirsjärven ja Hurmeen (2009) kuvaamat äänimateriaalin purkamiseen ja litterointiin liittyvät haasteet kuten puhujien identifiointi.

Haastattelut toteutettiin ryhmä- ja yksilöhaastatteluina marraskuun 2009 aikana. Haastatteluja varten määritettiin etukäteen sarja kysymyksiä, joilla ohjattiin keskusteluja oikeaan suuntaan. Kaikkia valmisteltuja kysymyksiä ei esitetty haastatteluissa, sillä osa kysymysten vastauksista saatiin vapaan

keskustelun kautta. Valmistellut kysymykset sekä keskustelujen kulku on dokumentoitu *liitteissä 2 ja 3*.

5.1.4 Haastattelujen tulokset

Toimeksiantajan sovellusalustaan kuuluvat tuotteet ovat Movenium Työajanseuranta (TAS), Movenium Ajopäiväkirja (APK) ja Movenium Kalustonhallinta (KH). Tuotteista on olemassa eri ominaisuuksilla varustettuja versioita, jotka on nimetty toisistaan eroavasti kuten Työajanseuranta Pro tai Työajanseuranta Pro RFID. Esimerkissä mainittu Työajanseuranta Pro RFID on ominaisuuksien avulla muokattu versio Movenium Työajanseuranta, jolla työajan merkitseminen voidaan tehdä RFID-tunnisteita lukemalla. Lyhenne RFID tulee sanoista Radio Frequency Indentificator. Tämä tarkoittaa etätunnistusta eli tiedon etälukua ja –tallentamista radiotaajuuksilla toimivien tunnisteiden avulla. RFID on heijastustekniikkaan perustuva tiedonsiirtomenetelmä (Seppä 2009, 6).

RFID-tunnisteen luku käsitetään kaupallisessa mielessä tuotteen teknisenä lisämodulina, jonka alle kuuluu tietty joukko sovellusalustan ominaisuuksia kuten se, mitä tunnisteen lukemisen jälkeen tapahtuu ohjelmassa. On huomattava, että tuotannon näkökulmasta katsottuna RFID-tunnisteiden luku on kuitenkin vain yksi sovellusalustan ominaisuus muiden joukossa, johon liittyy muita ominaisuuksia. Kaupallisen puolen tekemää yksiselitteistä tuote- tai moduulihierarkiaa ei ole olemassa sovellustasolla. Sen pitäisi kuitenkin olla tallennettuna tietokantaan. Sovellusalustan ominaisuus voi siis kaupallisessa mielessä tarkoittaa kokonaista tuotetta, tuotteen teknistä laajennusta eli moduulia tai yksittäistä asetusta tai ominaisuutta, joka voi liittyä tuotteeseen, moduuliin tai toiseen ominaisuuteen.

Sovellusalustan ominaisuuksien hinnoittelu perustuu kahteen pääperiaatteeseen: kuukausikohtaiseen palvelun käyttömaksuun tai käyttäjämäärään perustuvaan kuukausimaksuun. Toisin sanoen ominaisuus voi maksaa asiakkaalle esimerkiksi 5 €/kk riippumatta käyttäjämäärästä tai 5 €/kk jokaista käyttäjää kohti.

Yrityksen nimi	**Perus kk-maksu**	**Käyttäjämäärä**	**Tilatun ominaisuuden hinnoittteluperuste**	**Lisähinta perusmaksuun**
Yritys X	200 €	15	20 €/kk	20 €/kk
Yritys Y	200 €	15	5 €/kk/käyttäjä	75 €/kk

TAULUKKO 1 *Ominaisuuden hinnoittelutavan vaikutus asiakkaan kuukausilaskutukseen. Käytetyt luvut ovat kuvitteellisia.*

Taulukossa 1 kuvatun hinnoittelumallin lisäksi ominaisuuksiin voidaan tarvittaessa soveltaa myös muita hinnoittelumalleja. Tällaisia voivat olla esimerkiksi asiakkaan työajanseurantaan kuukauden aikana merkittyjen tuntien määrä, jolloin ominaisuuden hinta nousee portaittain sen mukaan, kuinka paljon merkintöjä on tehty. Tämän perusteella toimeksiantaja soveltaa laskutuksessaan sekä kiinteän veloituksen että transaktioihin perus-

tuvaa hinnoittelumallia. Ominaisuuden hintatietoihin on pystyttävä määrittelemään hinnoitteluportaiden rajat ja hinnat.

Nykyisen tietokannan rakenteesta ja nimeämiskäytännöistä selvisi, että rakenne on hyvin vähän säädelty. Yhteisesti sovittuja tai dokumentoituja nimeämiskäytäntöjä ei ole, tosin yrityksen sisällä on käytössä joitakin vakiintuneita tapoja. Kantataulut nimetään yleensä etuliitteellä *leimaus_*, *ajo_* tai *kalusto_* kyseessä olevan tuotteen mukaan. Jokaisen taulun perusavainkenttä on nimeltään *id*. Poikkeuksena tästä on taulu *leimaus_users*, jossa perusavainkenttä on nimeltään *primary_key*. Taulujen ja kenttien nimeämisessä pyritään käyttämään englannin kielisiä vastineita. Lähes joka taulusta löytyy kenttä *partnerid* tai *partner_id*, joka on viiteavain taulun *leimaus_partner* perusavaimeen. Tähän tauluun on tallennettu asiakkaiden tiedot.

Tuotekehitysosastolle nykyisen tietokannan rakenne aiheuttaa eniten ongelmia juuri sen takia, että käytäntöjä ja dokumentointia ei ole olemassa. Lisäksi tietokannassa on yksittäisten asiakkaiden tarpeiden mukaan tehtyjä muokkauksia, joiden tulee kuitenkin toimia kaikkien asiakkaiden käytössä yhtenevästi. Tuotekehitysjohtaja Veikko Salmisen haastattelun aikana virisi idea lisätä ominaisuuden tietoon myös päivämäärä, milloin tuotekehityksestä valmistuva ominaisuus on tuotantokelpoinen. Tämä olisi tärkeä tieto esimerkiksi myyjille, jotka pystyisivät tämän tiedon pohjalta kertomaan asiakkaille, milloin tuotekehityksen alla olevasta ominaisuus on saatavissa tai milloin sitä voidaan myydä.

5.1.5 Kyselyiden luonti

Johdon haastatteluissa selvisi, että toimeksiantajan strategian kannalta on tärkeää kerätä sovellusalustan ominaisuuksista vain tärkeäksi koettua tietoa, joka mahdollistaa helpon tiedon ja myytävien tuotteiden räätälöinnin asiakaskohtaisesti. Tällöin on mahdollista toteuttaa tehokasta SaaS-mallin mukaista tuotekauppaa sen sijaan että yritys pyrkisi myymään yksittäisiä projekteja. Henkilöstön työssään tärkeiksi kokemia tietoja voitiin kerätä kyselyllä.

Kyselyiden tarkoituksena oli hankkia sekä laadullista että määrällistä aineistoa. Vaikka kyselyyn osallistuvien henkilöiden määrä oli liian pieni tilastollisesti pätevien tulosten saamiseksi, voitiin kyselyllä saada tietoon sellaisia yleisiä käsityksiä ja tietoja, jotka eivät tulleet esille haastattelujen yhteydessä.

Tutkimuksen tiedonkeruun piiriin pystyttiin kyselyllä liittämään perustasolla koko toimeksiantajan henkilöstö. Vaikka saatujen tietojen sopivuudesta työn aineistoksi ei ollut varmuutta, perusteluna kyselyn suorittamiseksi voitiin pitää myös sitä, että kyselyllä pystyttiin tarjoamaan yrityksen johdolle tietoa henkilöstön käsityksistä ja tietotasosta sovellusalustan ominaisuuksiin liittyen. Tällaista tietoa ei ollut aiemmin kerätty.

Kyselyä varten luotiin kyselykaavake ohjeineen. Lomake suunniteltiin ja ohjeistettiin niin, että se oli mahdollista palauttaa sähköisesti tai paperisena. Käytetty kyselykaavake on merkitty *liitteeseen 1*. Kyselylomakkeen ohjeistusta ja kysymyksiä tarkennettiin ennen kyselyä pidettyjen haastattelujen perusteella. Haastattelujen yhteydessä kävi ilmi, etteivät kaikki yrityksen sisällä käyttäneet sovellusalustan ominaisuuksista tai niihin liittyvistä kaupallisista tuotteista samoja termejä, joten kyselylomake olisi ollut mahdollista tulkita väärin tai sen vaikeaselkoisuus olisi voinut heikentää ihmisten vastausmotivaatiota huomattavasti.

5.1.6 Kyselyiden tulokset

Kysely lähetettiin sähköpostilla koko toimeksiantajan henkilöstölle, joka käsitti 15 henkilöä. Kyselyyn vastasi kuusi henkilöä, jolloin vastausprosentti oli 40 %. Kaksi kyselyä palautettiin paperilla ja loput neljä sähköisenä. Vastaaminen oli mahdollista tehdä nimettömänä, mutta kaikki vastanneet palauttivat kyselyn omalla nimellään.

Kyllä	5
Ei	1

TAULUKKO 2 *Vastausjakauma: Tarvitsetko työssäsi tietoa siitä, minkälaisia ominaisuuksia sovellusalustaan kuuluu? (Kyllä/Ei)*

Suomenkielinen nimi (esim. Ajopk - vanha kilometrikorvaus)	*5*
Koodissa käytetty nimi (esim. ajo_old_km_comps)	*2*
Lyhyt kuvaus käyttötarkoituksesta	*6*
Käyttöohje	*2*
Mihin tuotteeseen tai tuotteen versioon (esim. Työajanseuranta Pro) ominaisuus kuuluu	*5*
Mitkä muut ominaisuudet liittyvät ominaisuuteen	*5*
Tekijä	*3*
Lisäyspäivämäärä	*3*
Ohjelmiston versio, jossa ominaisuus on ensimmäisen kerran ollut mukana	*2*
Hinta	*4*
Millä asiakkailla ominaisuus on käytössä	*2*
Missä tiedostoissa ominaisuutta käytetään	*2*

Pystyykö asiakas muuttamaan ominaisuutta (asetusta) omasta palvelustaan	*6*

TAULUKKO 3 *Vastausjakauma: Mitä tietoja ohjelmistoalustan ominaisuuksista tarvitsisit omassa työssäsi?*

Suomenkielinen nimi (esim. Ajopk - vanha kilometrikorvaus)	*0*
Koodissa käytetty nimi (esim. ajo_old_km_comps)	*0*
Lyhyt kuvaus käyttötarkoituksesta	*5*
Käyttöohje	*3*
Mihin tuotteeseen tai tuotteen versioon (esim. Työajanseuranta Pro) ominaisuus kuuluu	*5*
Mitkä muut ominaisuudet liittyvät ominaisuuteen	*3*
Tekijä	*1*
Lisäyspäivämäärä	*0*
Ohjelmiston versio, jossa ominaisuus on ensimmäisen kerran ollut mukana	*1*
Hinta	*2*
Millä asiakkailla ominaisuus on käytössä	*0*
Missä tiedostoissa ominaisuutta käytetään	*0*
Pystyykö asiakas muuttamaan ominaisuutta (asetusta) omasta palvelustaan	*1*
En tiedä / en osaa sanoa	*0*
Ei mitään	*0*

TAULUKKO 4 *Vastausjakauma: Mihin ominaisuuksiin liittyviin tietoihin kaipaisit eniten tarkennusta?*

Koska vastausprosentti jäi alle viidenkymmenen, ei tuloksista voitu johtaa täysin yksiselitteisiä johtopäätöksiä. Vastaukset antoivat kuitenkin suuntaa antavaa tietoa siitä, mitkä tiedot ominaisuuksista ovat tärkeitä yrityksen toiminnan kannalta. Yllättävää kyllä, kaupallisen henkilöstön ja teknisen tuotantohenkilöstön vastausten välillä ei ollut suuria eroja.

Lähes kaikki vastaajat olivat sitä mieltä, että he tarvitsevat tietoa sovellusalustan ominaisuuksista työssään. Vastausjakauma on esitetty *taulukossa 2*. Myös ainoa vastaaja joka ei mielestään tarvinnut näitä tietoja, oli kui-

tenkin antanut kehitysehdotuksia, joten hänelläkin oli jonkinlainen mielikuva ominaisuuksien tietojen puutteista.

Oman työn kannalta tärkeimmiksi koettiin ominaisuuden lyhyt kuvaus käyttötarkoituksesta sekä se, pystyykö asiakas muuttamaan ominaisuutta omasta palvelustaan käsin, kuten *taulukossa 3* on esitetty. Heti seuraavina listalla olivat ominaisuuden suomenkielinen nimi, mihin tuotteeseen tai tuotteen versioon ominaisuus kuuluu ja mitkä muut ominaisuudet siihen liittyvät. Eniten tarkennuksia toivottiin ominaisuuden lyhyeen kuvaukseen ja siihen, mihin tuotteeseen tai tuotteen versioon ominaisuus kuuluu. Vastausjakaumat on esitetty *taulukossa 4*.

Sanallisten vastauksien joukosta löytyi hyviä havaintoja ja parannusehdotuksia ominaisuuksien hallintaan liittyen. Dokumentaatio ja niiden muutoshistoria tulisi saada talteen. Kaupallinen rakenne ja hintatiedot haluttaisiin mukaan ominaisuuksien tietoihin samoin kuin ominaisuuksien keskinäiset riippuvuudet. Lisäksi vastauksissa annettiin parannusehdotuksia sovellusalustan käyttöliittymään. Eri käyttäjätasoihin liittyvät ominaisuudet haluttaisiin helposti haettaviksi ja esitettiin myös kysymys, onko tarpeen muokata yksittäisiä ominaisuuksia. vastaajan mieletä voisi ollaa mielekkäämpää muokata isompia kokonaisuuksia eli moduuleja.

5.1.7 Löydetyt puutteet

Tietokantaan ja ominaisuuksien tietoihin liittyvät puutteet voidaan jakaa kahteen ryhmään: itse tietokantaan liittyviin puutteisiin ja toimeksiantajan käytäntöihin tai sovellusallustaan liittyviin puutteisiin. Tietokantaan liittyviin puutteisiin on esitetty ratkaisuja luvussa 5.*2 Tietokannan jatkokehittäminen*. Yrityksen käytäntöihin ja sovellusalustaan liittyviin puutteisiin on annettu kehitysehdotuksia luvussa *6 Ehdotukset jatkotoimenpiteiksi.*

Tietokantaan liittyviä puutteita löytyi useita. Tietokannasta puuttuu tarpeellisia kenttiä. Joidenkin kenttien tyyppi on määritetty huonosti, joka mahdollistaa väärän tiedon tallentamisen. Ominaisuuksien keskinäisiä suhteita ei voida tallentaa muuten kuin henkilöstön antamina kommentteina. Ominaisuuksille tietokantaan tallennettavien hintatietojen rakenne ei vastaa kaupallista rakennetta. Kaikki ominaisuudet ovat tasa-arvoisia. Ominaisuuksien välillä ei ole tietokannassa hierarkiaa. Tietokannan tauluissa on käyttämättömiä tai vanhentuneita kenttiä. Ominaisuuksien ja käyttäjätasojen välisiä yhteyksiä ei tallenneta tietokantaan. Tietokannasta ei nähdä, voiko asiakas muuttaa ominaisuuden tilaa omasta palvelustaan vai onko tämä mahdollista ainoastaan toimeksiantajalle.

Puutteita löytyi myös toimeksiantajan käytännöistä ja sovellusalustasta. Tarpeellisia tietoja ominaisuuksista ei ole pakko antaa niitä luodessa. Ominaisuuksia voidaan lisätä sekä sovellusalustan koodin että käyttöliittymän kautta, mutta kumpikaan ei pakota antamaan riittävästi tietoja. Luomiskäytäntö vaihtelee henkilön mukaan. Ominaisuuksia ei dokumentoida tarpeeksi tarkasti. Muutoshistoria ei jää talteen. Kannan rakenteelle ei ole ollut selkeitä nimeämiskäytäntöjä, ainoastaan yleisiä suuntalinjoja.

Uusien ominaisuuksien toimintaa ei aina ole suunniteltu etukäteen tarpeeksi tarkasti. Ominaisuuksien hallinnan käyttöliittymä on puutteellinen. Jotkin ominaisuudet vaativat lisäasetuksia muualla sovellusalustan käyttöliittymässä kuin siellä, missä ominaisuuksia todellisuudessa hallitaan.

5.2 Tietokannan jatkokehittäminen

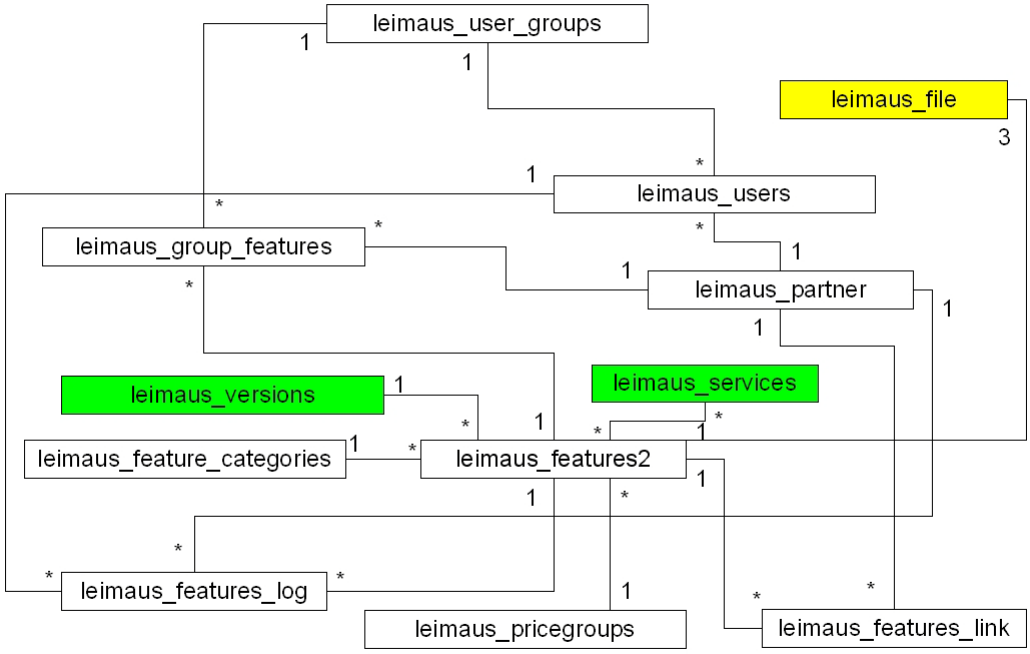
Kerätyn aineiston pohjalta on tässä luvussa annettu kehitysehdotuksia tietokannan rakenteeseen, joka on esitetty luvussa *5.1.1 Nykyisen tietokannan kartoitus*. Sovellusalustan toimintaan ja toimeksiantajan toimintamalleihin liittyviä kehitysehdotuksia on annettu luvussa 6 *Ehdotukset jatkotoimenpiteiksi*.

Tietokantaratkaisussa on pyritty ensisijaisesti käyttämään sovellusalustan vakiintuneita käytäntöjä, vaikka ne eivät olisikaan oikeaoppisen ja normaalisääntöjen mukaisen tietokantasuunnittelun kannalta täysin perusteltuja. Annetut kehitysehdotukset on tehty linjassa *luvussa 5.3 Aineiston kerääminen* saatujen tietojen perusteella. Näiden lisäksi pääperiaatteena on pidetty sitä, että perusavain kenttien lisäksi mitään muita taulujen kenttiä ei määritetä uniikeiksi, sillä tämä näytti olevan perusperiaate koko sovellusalustan sisällä. Mahdolliset hakujen ja tallennusten samanlaisten arvojen tarkistukset tehdään ohjelmallisesti.

5.2.1 Rakennemuutokset

Uusien tietokannan taulujen kentät ja niiden tyypit on esitetty *taulukoissa 5 ja 6*. Jotta ominaisuuksien jaottelu asiakkaille tarjottavien eri ominaisuuksia sisältävien palvelukokonaisuuksien mukaan olisi mahdollista, tulisi tarjottavien palvelujen löytyä tietokannasta. Taulun nimeksi tulisi *leimaus_services*. Näin *leimaus_feature2* taulun ominaisuudet voitaisiin linkittää viiteavaimella palvelukokonaisuuksiin kuten *kuvassa 3* on esitetty. Taulujen välinen yhteys on monesta moneen tyyppinen, koska ominaisuus voi kuulua moneen palveluun ja palvelulla on useita eri ominaisuuksia. Sovellusalustan versiot tulisi erottaa omaan kantatauluunsa *leimaus_versions* kuten *kuvassa 3* on esitetty. Ominaisuus linkitettäisiin viiteavaimella versioon, jossa se on tullut mukaan sovellusalustaan.

Sovellusalustassa oli jo taulu *leimaus_file*, jonne voidaan tallentaa sovellusalustaan liittyvien tiedostojen tietoja. Tämä taulu linkitetään ehdotuksessa *leimaus_features2* tauluun dokumentaatioiden liittämiseksi ominaisuuksiin. Ominaisuuksien dokumentointi ja dokumentoinnin versiointi voidaan hoitaa erillisissä tiedostoissa, jolloin pelkkä linkkitieto tietokannassa riittää tiedon hakemiseen käyttöliittymässä.



KUVA 3 *Ominaisuuksien hallintaan liittyvät kantataulut ja niiden suhteet ehdotetussa kantaratkaisussa. Lisätyt uudet taulut on merkitty vihreällä ja olemassa olleet ja ehdotuksessa rakenteeseen lisätyt taulut on merkitty keltaisella.*

Kenttä	**Tyyppi**	**Kommentti**
id	int(11)	Perusavain
name	varchar(250)	Palvelun nimi, esim. ”Työajanseuranta Pro”
created	date	Päivämäärä jolloin palvelu on lisätty.
comment	text	Palvelun kommentti.

TAULUKKO 5 *Taulun leimaus_services kentät.*

Kenttä	Tyyppi	Kommentti
id	int(11)	Perusavain
version	varchar(50)	Sovellusalustan versionumero.
created	date	Päivämäärä jolloin versio on lisätty tietokantaan.
published	date	Päivämäärä, jolloin version on julkaistu.
comment	text	Version kommentti.

TAULUKKO 6 *Taulun leimaus_versions kentät.*

5.2.2 Taulujen muutokset

Taulukossa 7 on esitetty tauluun *leimaus_features2* lisättävät kentät, niiden tietotyypit sekä kuvaus käyttötarkoituksesta. Kenttä *pricetype* liittyy ominaisuuden hinnoittelutiedon tallentamiseen. Tähän enum-tyyppiseen kenttään tallennetaan se hinnoitteluperuste, jota ominaisuuteen sovelletaan. Toimeksiantajan käyttämät hinnoitteluperusteet on käyty läpi luvussa *5.3.3. Haastattelujen tulokset*. *Taulukossa 7* on annettu *pricetype* kentälle kaksi mahdollista arvoa: month tai month_user, jotka kuvastavat kuukausikohtaista palvelun käyttömaksua ja käyttäjämäärään perustuvaa kuukausimaksua. Kentän mahdolliset arvot on helppo lukea käyttöliittymään. Kenttä voidaan jättää tallentaessa myös tyhjäksi, koska kaikki ominaisuudet eivät ole maksullisia.

Ominaisuuksien hintatiedot saadaan näin tallennettua käyttämällä kahden kentän yhdistelmää. Euromääräinen hinta saadaan kentästä *pricegroupid*, joka on viiteavain tauluun *leimaus_pricegroups*. Laskutusperuste taas saadaan *pricetype* kentästä.

Kenttä *serviceids* on tekstimuotoinen kenttä, johon on pilkulla eroteltu kaikki ne *leimaus_services* taulusta löytyvien palvelujen perusavaimet, joihin ominaisuus liittyy. Ratkaisu ei ole normaalisääntöjen mukainen,

mutta tapa noudattaa sovellusalustan käytäntöä monesta-moneen suhteissa. Pilkulla erotetut perusavaimet on helppo lukea ohjelmallisesti, eikä erillinen linkkitaulu näiden kahden taulun välillä poistaisi varsinaista ongelmaa. Viitattavia arvoja ei välttämättä ole yhtään tai niitä voi olla yhtä monta kuin *leimaus_services* taulussa on rivejä. Tosin voidaan sopia, että tauluun tallennetaan jokin tietty arvo, mikäli ominaisuus koskee kaikkia tarjottavia palveluja.

Ominaisuuksien dokumentaatiota ei ole järkevää tallentaa suoraan tietokantaan. Useimpien ominaisuuksien dokumentaatio on voitava esittää niin tuotantohenkilöstön, kaupallisen henkilöstön kuin asiakkaidenkin suuntaan sisällöltään erilaisina versioina ja lähes poikkeuksetta tehty dokumentaatio on erillinen tiedosto, jota päivitetään tarpeen vaatiessa. Tämän vuoksi tauluun *leimaus_features2* on lisätty kolme viiteavainta tauluun *leimaus_file*. Kyseinen tietokantataulu on jo olemassa, mutta sitä ei ole käytetty ominaisuuksien tietojen hallinnassa.

Lisättävät viiteavaimet *leimaus_file* tauluun ovat kenttä *production_manual_fileid*, joka viittaa tuotantohenkilöstön dokumentaatioon, *sales_manual_fileid* joka viittaa kaupallisen henkilöstön dokumentaatioon ja *customer_manual_fileid* joka viittaa asiakkaan dokumentaatioon. Ratkaisussa oletetaan, että dokumentaatiotiedostoja on vain yksi jokaista kohderyhmää kohti. Dokumentaation muutoshistoriaa voidaan pitää yllä dokumenttien sisällä.

Luettelotyyppisessä kentässä *service_type* määritetään ominaisuuden sijoittuminen kaupalliseen hierarkiaan. Valittavissa olevissa arvoilla *service* eli palvelu, *module*, eli moduuli ja *feature* eli ominaisuus määritetään, onko ominaisuus palvelukokonaisuus kuten ajopäiväkirja, palvelukokonaisuuden osakokonaisuus eli moduli kuten ajopäiväkirjan kilometrikorvaukset vai yksittäinen ominaisuus. Jaottelu on selvitetty luvussa *5.1.4 Haastattelujen tulokset.* Kenttä *featureids* toimii samoin kuin kenttä *serviceids.* Kenttään tallennetaan pilkulla eroteltuna kaikki samasta *leimaus_features2* taulusta löytyvien ominaisuuksien perusavaimet, joihin kyseinen ominaisuus liittyy.

Uutena mahdollisuutena ominaisuudelle voidaan asettaa tila kenttään *status*. Käytäntö on yleinen sovellusalustassa, jossa esimerkiksi käyttäjiä voidaan aktivoida ja passivoida. Passiivisten käyttäjien tietoja ei poisteta kokonaan kannasta. Passiivisten käyttäjien tietoja ei huomioida ohjelman toiminnassa. Esitetyssä ratkaisussa aktiivinen ominaisuus on täysin muokattavissa ja huomioidaan sovellusalustan koodissa. Passiivinen ominaisuus ei ole muokattavissa, eikä sen vaikutusta huomioida koodissa. Passiivinen ominaisuus voidaan kuitenkin palauttaa aktiiviseksi suoraan käyttöliittymästä. Vanhentuneen ominaisuuden tietoja ei ole tarkoitus poistaa lopullisesti tietokannasta. Se merkitään poistetuksi, jolloin ominaisuus ei näy käyttöliittymässä. Tilojen pohjalle voidaan rakentaa erilaisia käyttöliittymäratkaisuja.

Ominaisuuden vaikutus eri käyttäjätasoihin tallennetaan *userlevels* kenttään. Käyttäjätasot voidaan ilmaista numeroina samoin kuin ne ilmaistaan taulussa *leimaus_users*. Numerot erotetaan toisistaan pilkulla.

Kyselyiden perusteella katsottiin aiheelliseksi lisätä ominaisuudelle tieto siitä, voiko asiakas muuttaa ominaisuuden arvoa omassa palvelussaan. Tähän käytetään enum-tyyppistä kenttää *partner_editable*. Valittavina arvoina ovat joko kyllä tai ei.

Nimi	**Tyyppi**	**Kuvaus**
pricetype	enum(’month’, ’month_user’)	Kenttä määrittää, mitä hinnoitteluperustetta ominaisuuteen sovelletaan.
serviceids	varchar(300)	Tekstikenttä, jossa on pilkulla erotettuina viiteavaimet leimaus_services tauluun.
production_manual_fileid	int (11)	Tuotantohenkilöstön dokumentaatio. Viiteavain leimaus_file tauluun.
sales_manual_fileid	int (11)	Kaupallisen henkilöstön dokumentaatio. Viiteavain leimaus_file tauluun.
customer_manual_fileid	int (11)	Asiakkaiden dokumentaatio. Viiteavain leimaus_file tauluun.
service_type	enum(’service’, ’module’, ’feature’)	Kaupallinen hierarkia: Määritetään, onko ominaisuus palvelukokonaisuus, palvelukokonaisuuden osakokonaisuus eli moduli vai yksittäinen ominaisuus.
featureids	varchar(300)	Muut ominaisuudet, jotka liittyvät ominaisuuteen. Tekstikenttä jossa on pilkulla erotettuina viiteavaimet leimaus_features2 tauluun.
status	enum(’active’, ’passive’, ’removed’)	Ominaisuuden tila: ominaisuus voidaan määrittää aktiiviseksi, passiiviseksi tai poistetuksi.
userlevels	varchar(20)	Käyttäjätasot, joita ominaisuus koskee. Käyttäjätasot luetellaan numeroina pilkulla eroteltuna esimerkiksi 1,3,5.
partner_editable	enum(’yes’, ’no’)	Voiko asiakas muuttaa ominaisuuden arvoa omassa palvelussaan.

TAULUKKO 7 *Tauluun leimaus_features2 lisättävät kentät.*

Taulusta *leimaus_features2* voidaan poistaa kokonaan kentät *category* ja *show_level*. Kenttä *category* voidaan korvata kokonaan kentällä *categoryid*. Tämä kenttä on viiteavain *leimaus_feature_categories* tauluun. Käyttämällä selkeää viiteavainta vapaamuotoisen tekstikentän sijaan ominaisuuden kategoriaksi ei voida tallentaa ylimääräisiä arvoja. Tämä helpottaa myös tietojen hakemista käyttöliittymään. Kenttä *show_level* on kerätyn tiedon perusteella ylimääräinen, jota ei oteta huomioon missään sovellusalustassa.

Näiden lisäksi taulusta *leimaus_features2* jo löytyvä kenttä *version* muutetaan tekstikentästä numerokentäksi ja sen nimeksi vaihdetaan *versionid*. Kenttä toimii viiteavaimena luvussa *5.2.1 Rakennemuutokset* esiteltyyn tauluun *leimaus_versions*. Kenttä *type* vaihdetaan tekstikentästä luettelotyyppiseksi. Mahdollisiksi arvoiksi tulevat tietokannassa jo esiintyvät arvot: *checkbox*, *select*, *number*, *text* ja *textarea*.

Tietokantaratkaisuun ei otettu mukaan mahdollisuutta merkitä ominaisuudelle päivämäärää, jolloin tuotekehityksestä valmistuva ominaisuus on tuotantokelpoinen. Tätä voidaan perustella sillä, että uuden yksittäisen ominaisuuden tai moduulin valmistumisaika mitataan useimmiten päivissä, eikä keskeneräisiä ominaisuuksia normaalisti päivitetä tuotantopalvelimelle. Teknisesti sama ratkaisu voitaisiin toteuttaa myös linkittämällä keskeneräinen ominaisuus *leimaus_versions* taulussa jo olevaan, mutta vielä julkaisemattomaan versioon. Jos version julkaisupäivä on tulevaisuudessa, ei ominaisuuskaan voi tällöin olla vielä tuotantokäytössä. Ratkaisussa ei myöskään ole huomioitu normaalimuotojen sääntöjä, mikäli säännöt eivät olleet linjassa jo vakiintuneiden sovellusalustan käytäntöjen tai toiminnan kanssa.

6 EHDOTUKSET JATKOTOIMENPITEIKSI

Kerätyn aineiston perusteella toimeksiantajan toimintaan ja sovellusalustan kehitykseen liittyen voidaan antaa parannusehdotuksia. Nämä ehdotukset koskevat yrityksen hallintoa ja sisäisiä käytäntöjä, minkä takia ne käsitellään erillään tietokantaratkaisusta.

Tietokantaan ei voitu tallentaa kaikkea tarvittavaa tietoa, mutta tämän lisäksi yrityksen sovellusalustan ominaisuuksiin liittyvät dokumentointikäytännöt havaittiin puutteelliseksi. Onkin suositeltavaa, että tuotantohenkilöstölle luodaan joko teknisesti tai hallinnollisesti sovellusalustan ominaisuuksiin liittyvä dokumentointikäytäntö, jota tulisi seurata. Käytäntöä voidaan keventää tekemällä dokumentaatio moduulitasolla, jolloin yksittäisiä ominaisuuksia ei välttämättä tarvitse dokumentoida. Kyselyiden perusteella erityisesti ominaisuuksien lyhyitä kuvauksia olisi syytä tarkentaa, koska ne näkyvät sovellusalustan käyttöliittymässä ominaisuuksien hallinnassa.

Yrityksen kaupallinen henkilöstö tarvitsee enemmän tietoa järjestelmän mahdollisuuksista ja rajoituksista kuin mitä sen on tällä hetkellä mahdollista saada. Dokumentointikäytännön luonnissa tulisi ottaa huomioon tämä näkökulma. Kerättyä materiaalia voitaisiin hyödyntää myynnissä, jolloin asiakkaille voitaisiin markkinoida lisäominaisuuksia. Tiedon jakamista myös toiseen suuntaan eli kaupalliselta puolelta tuotantoon tulisi lisätä, jotta näkemykset asiakastarpeista olisivat ajan tasalla.

Tuotannon selkiyttämiseksi yrityksen olisi syytä luoda perustason tekninen nimeämiskäytäntö sekä tietokantaan että sovellusalustaan. Tällä voitaisiin nopeuttaa ohjelmistokehitystä rutiinitehtävissä, helpottaa virhetarkistusta ja tiedon jakamista sekä tulevaisuudessa yksinkertaistaa uuden tuotantohenkilöstön kouluttamista. Jos tämä ei ole mahdollista toteuttaa nykyiseen järjestelmään, asia tulisi ottaa huomioon tulevaisuudessa.

Sovellusalustassa olisi syytä kiinnittää huomiota ominaisuuksien hallintaan käyttöliittymässä sekä ominaisuusmuutosten lokitietojen tarkentamiseen. Ominaisuuksien hallintaan on olemassa kaksi erilaista lomaketta käyttöliittymässä, joista kumpikaan ei täytä kaikkia tarpeita. Lisäksi sovellusalustaa hallinnoivat käyttäjät pystyvät muokkaamaan ominaisuuksien asiakaskohtaisia arvoja suoraan asiakkaiden palveluissa. Ominaisuusmuutoksista tallennetut lokitiedot eivät kaikissa tapauksissa ole riittäviä tai käyttöliittymässä ei näytetä tarpeeksi tarkkoja tietoja.

Työn valmistumisen aikana toimeksiantaja on jo selkiyttänyt sovellusalustan versiointia niin, että versiokohtainen ominaisuusluettelo on mahdollista tehdä jakelua varten. Kerättyä tietoa versioista ja niiden julkaisussa mukana tulevista ominaisuuksista voitaisiin edelleen tarkentaa. Asia voitaisiin sisällyttää dokumentointikäytäntöihin.

7 YHTEENVETO

Olemassa olevan tietokannan kartoitus onnistui hyvin ottaen huomioon dokumentaation puutteen ja tietokannan monimutkaisuuden. Joiltakin osin kartoituksen tuloksia ei voida pitää täysin luotettavina. Tietokantarakenteessa on osia, joiden alkuperäisestä tarkoituksesta ei voitu päätellä mitään sovellusalustan koodin perusteella, koska niitä ei enää käytetty. Näissä tapauksissa jouduttiin turvautumaan henkilöstön tietoihin tai muistikuviin vanhoista käyttötarkoituksista. Järjestelmän tutkiminen paljasti hyvin dokumentoinnin puutteita, mutta myös joustavan työskentelytavan edut sovelluskehityksessä.

Haastattelujen tekstimuotoisesta sisällöstä oli helppo purkaa auki työn kannalta olennaiset tiedot. Haastattelujen avulla kerätyt tiedot olivat erittäin tarpeellisia. Ne tukivat erityisesti sovellusalustan ominaisuuksiin liittyvän kaupallisen rakenteen selvitystä sekä olemassa olevan dokumentaation tason määrittämistä. Erityisen positiivinen kokemus oli sekin, että haastattelujen yhteydessä myös haastateltavat saivat uusia tietoja.

Kyselyiden aineisto oli laadukasta erityisesti sanallisten vastausten osalta. Tilastollisesti kyselyiden arvo jäi kuitenkin vähäiseksi. Syynä olivat alhainen vastausprosentti sekä alun perinkin pieni mahdollisten vastaajien määrä. Kyselyt antoivat tietoa siitä, mitä tietoja toimeksiantajan henkilöstö koki tärkeäksi ja millä tavoilla he haluaisivat parantaa sovellusalustan tietojen hallintaa. Jälkikäteen ajatellen kyselyiden arvo oli suurempi toimeksiantajan toiminnan kehittämiselle ja dokumentaation kehittämiselle kuin itse tietokantaratkaisulle.

Kokonaisuutena tutkimuskysymyksiin vastaaminen onnistui hyvin. Työn valmistumisen aikana toimeksiantajalla tehtiin sovellusalustaan liittyvää tuotekehitystä, joka tulee pitkällä aikavälillä muuttamaan nykyisen ominaisuusrakenteen täysin. Tämä ei kuitenkaan tarkoita sitä, ettei työn tuloksia voitaisi käyttää hyväksi tulevaisuudessa, mutta se laski jossain määrin motivaatiota. Tulosten valmistuessa oli aiheellista epäillä myös toimeksiantajan resurssien riittävyyttä kuvattujen tietokantamuutosten tekemiseen.

Työn tekeminen antoi runsaasti tietoa SaaS-mallista, joka oppimiskokemuksena oli antoisin. Työelämän kannalta opinnäytetyö opetti dokumentoinnin ja tiedon jakamisen haasteellisuuden. Laajan ohjelmistokehityksen kurinalaisuus osoittautui tärkeäksi ja todennäköisesti toimeksiantajan kannalta tärkeimmäksi yksittäiseksi asiaksi, josta on syytä pitää kiinni.

LÄHTEET

Hirsjärvi Sirkka & Hurme Helena 2009. Tutkimushaastattelu: teemahaastattelun teoria ja käytäntö

Hovi Ari, Huotari Jouni, Lahdenmäki Tapio, 2003. Tietokantojen suunnittelu & indeksointi

Seppä Heikki, Tekesin katsaus 249/2009. Vallankumouksellinen RFID, Etätunnisteteknologian kehitys meillä ja maailmalla.

SIIA, Software & Information Industry Association, 2001.
Software as A Service: Strategic Backgrounder
http://www.spa.org/estore/pubs/SSB-01.pdf

Donaldson, Sonya A. Techwatch 2009. Add A Little SaaS to Your Business

PHP ohjelmointikielen virallinen manuaali, http://php.net. Viitattu 07.11.2009. http://fi.php.net/manual/

MySQL:n virallinen dokumentaatio, http://www.mysql.com. Viitattu 07.11.2009. http://dev.mysql.com/doc/

Skype Limited, Viitattu 5.12.2009. http://about.skype.com/

Mantis Bug Tracker. Viitattu 23.1.2010. http://www.mantisbt.org

The Apache Software Foundation. Viitattu 07.11.2009.
http://www.apache.org/

MUU TAUSTAMATERIAALI

Frey Kelly L. Sr., Hall Thomas J., 2007, Application Service Provider and Software As a Service Agreements Line by Line - A Detailed Look at ASP and SaaS Agreements and How to Change Them to Meet Your Needs

Jyrinki, Erkki 1976. Kysely ja haastattelu tutkimuksessa

Chaffey, Dave 2009. E-Business and E-Commerce Management: Strategy, Implementation and Practice

Laudon, Ken, Laudon, Jane 2009. Management information Systems

Parantainen, Jari 2007. Tuotteistaminen

Shein Esther, Computerworld. Vol. 42, no. 39, pp. 26-27. 29 Sept. 2008. Saved by SaaS

LeCayla Technologies. A Whitepaper for Independent Software Vendors (ISVs) - Properly Pricing Software-as-a-Service Within an Established Product Portfolio
http://thinkstrategies.icentera.com/exLink.asp?5116660OP15G19I25355330

Kolehmainen Aleksi, Tietoviikko-lehti, Tivi.fi, 13.3.2009. Ohjelmistoja-palveluina? Muista nämä kysymykset.
http://www.tietoviikko.fi/kaikki_uutiset/article252685.ece

LIITE 1

Kysely ohjelmistoalustan ominaisuuksista

LIITE 2

Ryhmähaastattelu

LIITE 3

Veikko Salmisen yksilöhaastattelu

LIITE 4

Tietokannan taulukuvaukset